Master Thesis Proposal

Evaluation of different tools for design and fault-injection of asynchronous circuits

Martin Schwendinger
Advisors: Dipl.-Ing. Dr.techn. Andreas Steininger
Dipl.-Ing. Florian Huemer

January 15, 2022

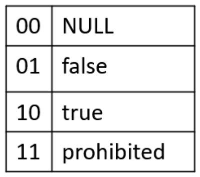
1 Problem Statement

In contrast to conventional synchronous circuits (SCs) there are no established standard tools for design and simulation of asynchronous circuits (ACs). For SCs tools are split up by their intended target. Established commercial design tools for FPGAs are e.g. *Intel Quartus Prime* and *ModelSim* or *Xilinx Vivado*. Prominent open source tools for mere RTL synthesis or simulation are e.g. *Yosys* and *GHDL*. For commercial ASIC design tools the offers of companies like *Synopsys*, *Cadence Design Systems*, *Siemens* or *Silvaco* are an option. For development of ACs often a mix of tools originally intended for SCs design is (mis)used, depending on personal preferences and target technology. Tools especially intended for ACs design are mostly prototypes usually of university origin.

More concrete, a very promising way of designing ACs is by NULL-Convention-Logic (NCL), see figure 1. NCL defines an acknowledge signal and (multiple) data signals, where one signal usually consists of two rails. So one logic bit is represented by two rails. Data synchronization is achieved by alternating between data phase, acknowledge toggles and NULL phase (data phase $\rightarrow$ acknowledge rises $\rightarrow$ NULL phase $\rightarrow$ acknowledge falls $\rightarrow$ next data phase). A circuit designed using NCL is often referred to as quasi delay insensitive (QDI) circuit. An alternative would be Bundled Data, where synchronization is achieved by a digital request and acknowledge signal, as well as some timing constraints about the data signals (each logic bit represented as one digital signal, as usual), see figure 2.

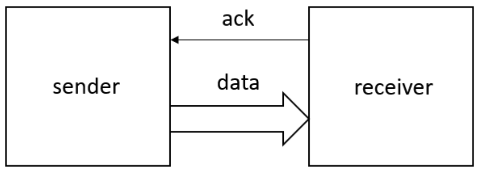
Hence the potential of a certain tool to establish itself in the asynchronous design domain is also very depending on its support for NCL or Bundled Data. For an individual tool lent from synchronous design flows, the minimum criterion is of course that it does not actually prevent profitable usage of these design styles. To check this criterion for a certain tool can already be a harsh task, because usually there are no official advices regarding ACs, if the tool is intended for SCs. In contrast, tools dedicated to ACs design and simulation are usually very focused to a concrete use case, which is maybe only made up to boost a specific research interest of its university associated developer.

When now aiming for fault injection tests, additionally to design tools also simulation tools with the functionality to force an arbitrary (internal) signal to inject the actual fault are needed.

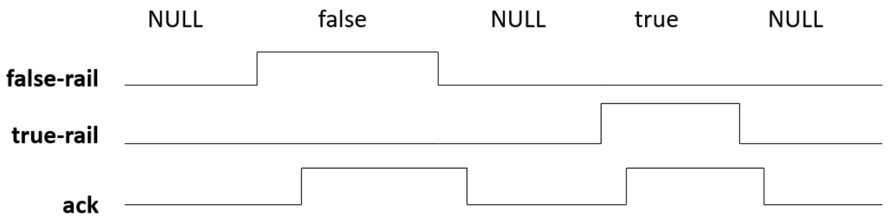


00	NULL
01	false
10	true
11	prohibited

(a) Data encoding for two rails representing one bit.

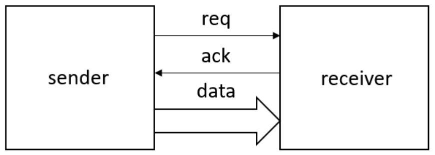


(b) Signal directions between sender and receiver.

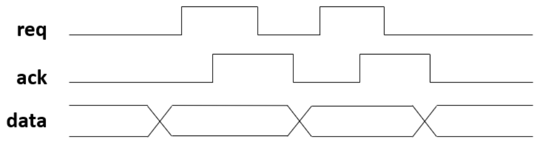


(c) Signal timing diagram.

Figure 1: NULL Convention Logic



(a) Signal directions between sender and receiver.



(b) Signal timing diagram.

Figure 2: Bundled Data

Conveniently for pure simulation there is a wider range of appropriate (open source) tools. Minimal criteria like a force option and catching signal transitions also in absence of a clock are common. The critical part here is to convert the peculiar circuit representation format from the AC tool to one accepted by the simulation tool chosen. So the pairing of a design tool with a simulation tool can be considered the first step to an actual design flow. Further steps would be the mapping to a technology and providing a layout to finally arrive at a GDSII representation.

Finally, even after arriving at GDSII format there still remain questions about convenience, performance and competitiveness. ACs are considered to be more performant (particularly faster, more energy efficient), nevertheless the willingness to invest in the concept is low, due to apprehension that initial cost to bring a AC to market will not pay off.

2 Expected Results

This thesis aims to present two distinct design flows for ACs design. One developed by the *Asynchronous VLSI and Architecture Group* at Yale University, the other one developed by the *Embedded Computing Systems (ECS) Group* at TU Wien. Both design flows feature a bunch of individual tools, some of which are open source, others are proprietary. For each design flow the previously raised questions about capability for (high level) asynchronous design, simulation (including fault injection opportunities) and technology mapping for actual hardware will be discussed and compared. Possible compatibility of individual tools of the two different design flows will be evaluated and even further enabled through specific development of intermediate tools. The goal is to come up with an integration of both design flows.

Expressed differently the following questions should be answered:

- What are the individual tools capable of?
- How do they satisfy their designated purpose in their respective design flow?
- What are the full capabilities of the two design flows? How do they compare to each other?
- How can the two design flows enhance or partly substitute each other?
- To which extent can the problems (from previous section) be solved by a integration of the original design flows?
- Which contribution to the actual fault injection research project by the ECS group at TU Wien is possible by the integrated design flow?

3 Methods

3.1 Literature research

The theory of asynchronous circuits dates back to the report by David E. Muller in 1955 [13]. From thereon the concept was always present in research in the shadow of synchronous design concept. Further interest in asynchronous circuits is usually justified by its potential to beat synchronous circuits in terms of performance. So after a brief overview of evolution of ACs concept and its claims, the thesis will introduce various references about the specific asynchronous design practices

NCL and Bundled Data and furthermore the papers describing the preexistent design flow it is based on. It is intended to review the progress of the *Asynchronous VLSI and Architecture Group* at Yale University and the ECS group TU Wien in the field of ACs a bit. Of course every other literature, which is sufficiently related to the discussed topic or suitable as a base for certain points, is also considered.

3.2 Evaluate capabilities of design flows

As both design flows feature quite many different tools, either self developed, open source or proprietary, it is surely beneficial to present each one and discuss its individual capabilities and background outside of the design flow. A examination of its purpose in the design flow will follow. Finally the two design flows will be compared, where first hints to mutual enhancements should arise.

3.3 Integrate both design flows to a common customizable design flow

The two design flows aim for quite different goals. The design flow presented by the Yale University aims to provide an industry competitive open source approach from high level design entries (CHP, Dataflow) to low level description of transistor physics (size, leakage) including layout and simulation/verification tools to arrive at a well tested GDSII representation. So it intends to be a real open source alternative to commercial tools in chip design. The design flow from the ECS group at TU Wien is more devoted to the current fault injection project. The ACs are described as production rule set generated by a python library. The production rule sets are encapsulated by modules, which have inputs and outputs and can be connected like usual in hardware description languages. Also all other steps like simulation or fault injections are orchestrated python scripts. For simulation a conversion to VHDL is featured. Then established tools like ModelSim (proprietary) or GHDL (open source) are used. The benefit drawn from this difference in goals is that the integrated design flow can aim for both goals with the specific focus determined by the custom chosen path trough it. However, this deviation in goals may also hold various limitations on how tools from distinct design flows are compatible. Ultimately it is aimed for a top level python script orchestrating all other tools.

3.4 Demonstration of capabilities by fault injection experiments

Due to the current fault injection project of the ECS group the use case to illustrate the functionality of both individual flows and the integrated design flow will be a fault injection experiment. Particularly already used fault injection schemes by the ECS group will be applied to the integrated flow. An analysis and comparison of the simulation procedure and its results follows. It is open which alternations or even advancements can be made to the existing fault injection scheme and how these new results will be put in context to a trivial same input same output expectation.

3.5 Conclusion

After reviewing literary descriptions of many different approaches for designing ACs and their featured tools, extensively study the design flows from the Yale University and TU Wien and comparing these two, presenting an enhanced version by integration of the two and providing a use case of fault injection experiments, finally a conclusion of overall capabilities and on all upsides and downsides of the design flow variations will be presented. A comparison to the theoretic goals stated in section one follows.

4 State of the Art

The first theoretical mention of ACs design was by David E. Muller in 1955 [13]. The famous book *Switching Theory* in 1965 by Raymond Miller[10] later included this theory. A book also containing a brief history of asynchronous circuit design is *Asynchronous Circuit Design* by Chris J. Myers [14]. In this brief history old chips like the ILLIAC, DDMP, MIPS R3000 are mentioned and that Philips Research Laboratories developed various ACs aiming for low power consumption (an error corrector for a cassette player, as well as a pager decoder). Recommended introductions to ACs are [19] or [20] both by Jens Sparsø. Furthermore over the years various overview papers like in two parts [15] and [16] or [6] have been published.

One field of application is development of neural network hardware. Two famous papers are out there. [5], where a neurogrid i.e. a neuromorphic system to simulate the function of a biological neural system in real time was developed, and [1], where a neurosynaptic processor was developed under participation of the *Asynchronous VLSI and Architecture Group* (Yale). Asynchronous circuits are very suitable for development of neural network hardware, because neural networks are event driven as there is no global clock in biologic systems. [9] claims that the difficult adoption possibilities of the design in [1] due to the lack of commercially available design tools was motivating for the open source design flow developed at Yale University. There is also an article [2] covering this design flow. Other recent publication by the Asynchronous VLSI and Architecture group at Yale University are e.g. [8], where a high level synthesis tool for asynchronous circuits from data flow is presented, or [21], where a gridded cell placer to tackle lack of physical layout automation tools for asynchronous circuits is presented.

TU Wien is currently focused on fault injection experiments in ACs. [3] presents the tools for generating the asynchronous circuit, to which then faults are injected and results analyzed. [4] also features fault injection experiments, this time more focused on analyzing the fault sensitivity of QDI pipeline circuits over various alternations (e.g. using an altered buffer or arithmetic unit in pipeline). The appealing property of fail stop behavior, which applies by using a (specific version of a) WHCB buffer is discussed in [18]. Some papers from other institutes about fault injections in asynchronous circuits are [11] (analyses fault sensitivity of QDI circuits), [7] (especially SEU events and possible tolerance against it), again related to the Yale University [17] (defines a concurrent failure detection method for pipelined asynchronous circuits) or [12] (examines usability of asynchronous logic for protection against fault injections).

Synoptically asynchronous circuits haven been around since mid-1950s. Since the past there are chips and other gadgets making use of it. Today, besides many papers providing repetitively an overview, reflect once more about the design style itself or testing it against some properties like robustness to fault injections, the most prominent application seems to be the development of neural networks.

5 Relevance to the Curriculum

This thesis aims to present, further enhance and integrate the design flows for asynchronous circuits from Yale University and TU Wien. So contents of the various courses related to digital design will be covered mainly. Fault injection experiments were also already an example especially in *Advanced Digital Design LU*. The then designed circuits will apply concepts covered in *Advanced Computer Architecture VU* as well. In general the topics covered are mainly related to the *Digital Circuits and Systems* module in the *Computer Engineering* curriculum. The most relevant courses are listed below:

- 182.700 VU HW/SW Codesign
- 182.701 LU HW/SW Codesign
- 182.754 LU Advanced Digital Design
- 182.755 VU Advanced Digital Design
- 182.756 VU Advanced FPGA Design
- 191.105 VU Advanced Computer Architecture

References

[1] F. Akopyan, J. Sawada, A. Cassidy, *et al.*, “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015. DOI: 10.1109/TCAD.2015.2474396.

[2] S. Ataei, W. Hua, Y. Yang, *et al.*, “An open-source eda flow for asynchronous logic,” *IEEE Design and Test*, vol. 38, pp. 1–10, Jan. 2021. DOI: 10.1109/MDAT.2021.3051334.

[3] P. Behal, F. Huemer, R. Najvirt, and A. Steininger, “An automated setup for large-scale simulation-based fault-injection experiments on asynchronous digital circuits,” in *2021 24th Euromicro Conference on Digital System Design (DSD)*, 2021, pp. 541–548. DOI: 10.1109/DSD53832.2021.00087.

[4] P. Behal, F. Huemer, R. Najvirt, A. Steininger, and Z. Tabassam, “Towards explaining the fault sensitivity of different qdi pipeline styles,” in *2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2021, pp. 25–33. DOI: 10.1109/ASYNC48570.2021.00012.

[5] B. V. Benjamin, P. Gao, E. McQuinn, *et al.*, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014. DOI: 10.1109/JPROC.2014.2313565.

[6] S. Hauck, “Asynchronous design methodologies: An overview,” *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69–93, 1995. DOI: 10.1109/5.362752.

[7] W. Jang and A. Martin, “Seu-tolerant qdi circuits [quasi delay-insensitive asynchronous circuits],” in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, 2005, pp. 156–165. DOI: 10.1109/ASYNC.2005.30.

[8] R. Li, L. Berkley, Y. Yang, and R. Manohar, “Fluid: An asynchronous high-level synthesis tool for complex program structures,” in *2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2021, pp. 1–8. DOI: 10.1109/ASYNC48570.2021.00009.

[9] R. Manohar, “An open-source design flow for asynchronous circuits,” *Government Microcircuit Applications and Critical Technology Conference*, Mar. 2019.

[10] R. E. Miller, *Switching theory.* John Wyley and Sons, 1965.

[11] Y. Monnet, M. Renaudin, and R. Leveugle, “Asynchronous circuits sensitivity to fault injection,” in *Proceedings. 10th IEEE International On-Line Testing Symposium*, 2004, pp. 121–126. DOI: 10.1109/OLT.2004.1319669.

[12] ——, “Designing resistant circuits against malicious faults injection using asynchronous logic,” *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1104–1115, 2006. DOI: 10.1109/TC.2006.143.

[13] D. E. Muller, “Theory of asynchronous circuits, report no. 66,” *Digital Computer Laboratory, University of Illinois at Urbana-Champaign*, 1955.

[14] C. J. Myers, *Asynchronous Circuit Design.* J. Wiley and Sons, 2001.

[15] S. M. Nowick and M. Singh, “Asynchronous design—part 1: Overview and recent advances,” *IEEE Design Test*, vol. 32, no. 3, pp. 5–18, 2015. DOI: 10.1109/MDAT.2015.2413759.

[16] ——, “Asynchronous design—part 2: Systems and methodologies,” *IEEE Design Test*, vol. 32, no. 3, pp. 19–28, 2015. DOI: `10.1109/MDAT.2015.2413757`.

[17] S. Peng and R. Manohar, “Efficient failure detection in pipelined asynchronous circuits,” in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’05)*, 2005, pp. 484–493. DOI: `10.1109/DFTVS.2005.33`.

[18] R. E. Shehaby and A. Steininger, “Analysis of state corruption caused by permanent faults in wchb-based quasi delay-insensitive pipelines,” in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2021, pp. 63–68. DOI: `10.1109/DDECS52668.2021.9417024`.

[19] J. Sparsø, “Asynchronous circuit design - a tutorial,” in *Chapters 1-8 in ”Principles of asynchronous circuit design - A systems Perspective”*, Boston / Dordrecht / London: Kluwer Academic Publishers, Dec. 2001, pp. 1–152. [Online]. Available: `%5Curl%7Bhttp://www2.compute.dtu.dk/pubdb/pubs/855-full.html%7D`.

[20] J. Sparsø, *Introduction to Asynchronous Circuit Design.* English. DTU Compute, Technical University of Denmark, 2020, Paperback edition available here: `https://www.amazon.com/dp/B08BF2PFLN`.

[21] Y. Yang, J. He, and R. Manohar, “Dali: A gridded cell placement flow,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.